Overview of SocialSim Program

The main method for the actual program is found in SocialSim so use this to run the program. Use -i by itself afterwards on the command line to run the program in interactive mode, or -s with <network_file> <event_file> <likeProbability> <followProbability> to run in simulation mode.

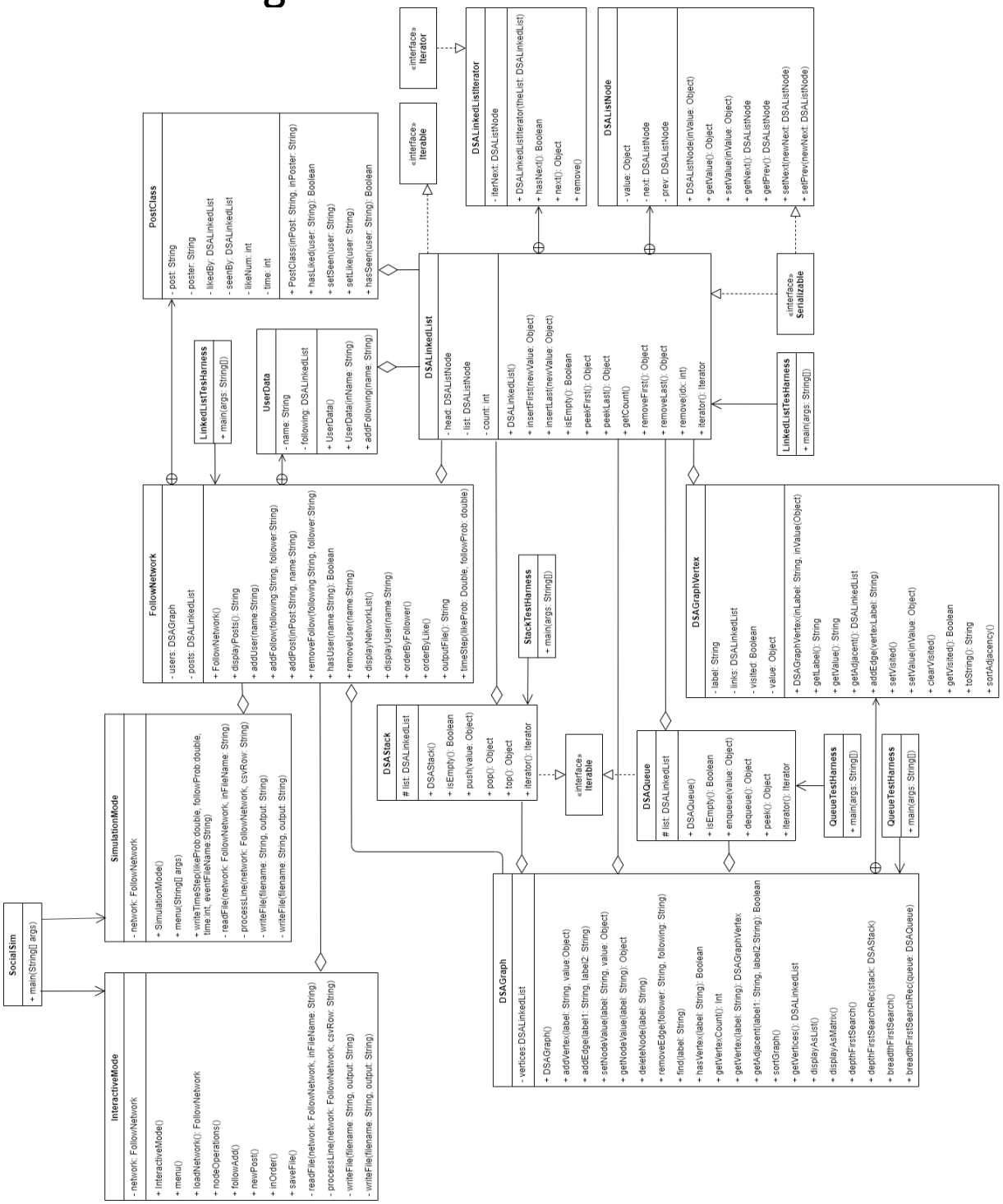
Interactive mode contains a big menu with lots of options. It can load a network file with a specified name into the program, can set the like and follow probabilities and display them. It can find/delete/insert a user inside the network, it can remove/add a follow, add a post, display the network as well as display users in order by amount of followers and also display posts in order of how many likes they have. The program menu also can save the network to a file and run a timestep. This file also contains the actual methods to print and read/process the files.

Simulation mode has a much smaller menu. It starts by reading from the command line the probabilities of following and liking and reads the network and event file given to it. The program then starts the menu in which you can either exit the program of write a timestep. If a timestep happens this program outputs the probabilities of follow/like and also the state of the network including users and posts.

The follow network contains the majority of the methods needed for the program to work correctly and contains all of the ADT's for the network. This file contains the functions to output all of the network information correctly and also add/remove/find users or posts.

Perhaps the most important method within this file is the timestep which is used by both simulation and interactive mode. The timestep is a propagation through the network. It starts by showing the post to each of the followers of the user that posts the post. They each have a probability to like the post, and if they do the post will be shown to their followers on the next timestep propagation. In the next propagation the users who have liked the posts show the post to their followers which in turn have a probability of liking and showing their followers. If they do like the post there is also a chance they will follow the user who posted the post. This doesn't happen on the first iteration as the first propagation is through the posters followers, so they are all already following the user.

UML Class Diagram



Descriptions of classes

Two of the classes with major functionality are the SimulationMode and InteractiveMode classes. These two classes contain functionality for both versions of the program. I chose two have two classes instead of one giant class as its separating their functionalities and makes it more easy to follow. They differ in their menu system primarily, where SimulationMode just does timestep while interactive

can add stuff and remove from the social network. Looking back now however, it would've been better to use inheritance to clean up the duplicate methods or have another class specifically for FileIO.

The social network itself is called FollowNetwork and contains the entire network inside of it. This contains all of the functionality to add/remove people and posts, print users/posts, sort, timestep and anything that requires the network.

Inside of the FollowNetwork there is a private inner class called UserData which is a user object that has all the information for the user such as name, and a list of who they are following. This didn't need to be another public class specifically as it doesn't need to be accessed outside of FollowNetwork so thus it is a private inner class of the Network.

Another private inner class inside of FollowNetwork is PostClass which is a post object that contains all of the information for a given post such as the name of the person who posted it, the number of likes it has, and the lists of who has liked it or seen it and also what "time" they posted the post. This also didn't need to be another public class specifically as it just needed to be accessed by the Network so it is a private inner class inside of the Network.

ADT Usage and Justification

The main ADT used for the Social network is a graph this is because a vertex can have as many children as possible unlike a tree. This could've been done with linked lists but the graph already has implementations of linked lists with the same functionality so a graph of users was the best option.

The graph vertexes, themselves are UserData objects which contain a name but also a Linked list that contains Strings with the names of the users that are following the user. Because the graph is directional to find who people are following you have to iterate through every vertex in the graph and their adjacency matrix to see if they are there. That is complicated and covers unnecessary time complexity so I chose to just have a linked list. I chose a linked list specifically compared to a queue or stack as it had a remove function as removing following could be in the middle of a list.

Another predominant ADT used in the network was another linked list that contains PostClass objects. I chose a linked list because it doesn't need to be ordered and is easy to iterate through especially for a timestep or other such method that needs to just go through one by one from beginning to end of the list.

Within the actual PostClass objects are two more linked lists. One is for storing the users that have liked a post, and the other is to store the users who have seen the post. I chose linked lists because they didn't need to be sorted like a tree and also the objects didn't need to relate to each other like a graph. A stack and queue wasn't used because the list will never encounter a time for remove, so it doesn't matter for removing from a particular side or inserting to a particular side. It just iterates from the beginning to the end of the list.